

## **AMENDMENTS TO THE CLAIMS**

This listing of claims will replace all prior versions, and listings, of claims in the application:

### **Listing of Claims:**

- 1           1. (Currently amended) A method for selectively monitoring load  
2 instructions to support transactional execution of a process, comprising:  
3           starting a transactional execution of a block of instructions in a program,  
4 wherein starting the transactional execution involves executing an explicit  
5 ~~performing an~~ instruction implemented in hardware to start the transactional  
6 execution;  
7           encountering a load instruction during the transactional execution, wherein  
8 changes made during the transactional execution are not committed to the  
9 architectural state of a processor until the transactional execution successfully  
10 completes;  
11           determining whether the load instruction is a monitored load instruction or  
12 an unmonitored load instruction by analyzing the load instruction;  
13           if the load instruction is a monitored load instruction,  
14                       performing a corresponding load operation, and  
15                       load-marking a cache line associated with the load  
16                       instruction to facilitate subsequent detection of an interfering data  
17                       access to the cache line from another process; and  
18           if the load instruction is an unmonitored load instruction, performing the  
19 corresponding load operation without load-marking the cache line.

1           2. (Original) The method of claim 1, wherein prior to executing the  
2 program, the method further comprises generating the instructions for the  
3 program, wherein generating the instructions involves:  
4           determining whether load operations that take place during transactional  
5 execution need to be monitored;  
6           generating monitored load instructions for load operations that need to be  
7 monitored; and  
8           generating unmonitored load instructions for load operations that do not  
9 need to be monitored.

1           3. (Original) The method of claim 2, wherein determining whether a load  
2 operation needs to be monitored can involve examining a data structure associated  
3 with the load operation to determine whether the data structure is a “protected”  
4 data structure for which loads need to be monitored, or an “unprotected” data  
5 structure for which loads do not need to be monitored.

1           4. (Original) The method of claim 2, wherein determining whether a load  
2 operation needs to be monitored can involve determining whether the load  
3 operation is directed to a heap, wherein loads from the heap need to be monitored  
4 and loads from outside the heap do not need to be monitored.

1           5. (Original) The method of claim 2, wherein determining whether a load  
2 operation needs to be monitored can involve allowing a programmer to determine  
3 if the load operation needs to be monitored.

1 |           6. (Currently amended) The method of claim 1, wherein determining  
2 whether the load instruction is a monitored load instruction involves examining an  
3 op code of the load instruction.

1           7. (Currently amended) The method of claim 1, wherein determining  
2 whether the load instruction is a monitored load instruction involves examining an  
3 address associated with the load instruction to determine whether the address falls  
4 within a range of addresses for which loads are monitored.

1           8. (Original) The method of claim 7, wherein examining the address  
2 involves comparing the address with one or more boundary registers.

1           9. (Original) The method of claim 7, wherein examining the address  
2 involves examining a Translation Lookaside Buffer (TLB) entry associated with  
3 the address.

1           10. (Original) The method of claim 1, wherein if an interfering data access  
2 from another process is encountered during transactional execution of the block of  
3 instructions, the method further comprises:  
4           discarding changes made during the transactional execution; and  
5           attempting to re-execute the block of instructions.

1           11. (Original) The method of claim 1, wherein if transactional execution of  
2 the block of instructions completes without encountering an interfering data  
3 access from another process, the method further comprises:  
4           committing changes made during the transactional execution to the  
5 architectural state of the processor; and  
6           resuming normal non-transactional execution of the program past the  
7 block of instructions.

1           12. (Original) The method of claim 1, wherein an interfering data access  
2 can include:

3           a store by another process to a cache line that has been load-marked by the  
4 process; and  
5           a load or a store by another process to a cache line that has been store-  
6 marked by the process.

1           13. (Original) The method of claim 1, wherein the cache line is load-  
2 marked in level 1 (L1) cache.

1           14. (Currently amended) An apparatus that selectively monitors load  
2 instructions to support transactional execution of a process, comprising:  
3           a start transactional execution mechanism configured to start a  
4 transactional execution of a block of instructions in a program, wherein starting  
5 the transactional execution involves executing an explicit performing an  
6 instruction implemented in hardware to start the transactional execution;  
7           an execution mechanism within a processor;  
8           wherein the execution mechanism is configured to support the  
9 transactional execution, and wherein changes made during the transactional  
10 execution are not committed to the architectural state of a processor until the  
11 transactional execution successfully completes;  
12           wherein upon encountering a load instruction during transactional  
13 execution, the execution mechanism is configured to,  
14                       determine whether the load instruction is a monitored load  
15                       instruction or an unmonitored load instruction by analyzing the  
16                       load instruction,  
17                       if the load instruction is a monitored load instruction, to  
18                       perform a corresponding load operation, and to load-mark a cache  
19                       line associated with the load instruction to facilitate subsequent

20 detection of an interfering data access to the cache line from  
21 another process; and  
22 if the load instruction is an unmonitored load instruction, to  
23 perform the corresponding load operation without load-marking  
24 the cache line.

1 15. (Original) The apparatus of claim 14, further comprising an instruction  
2 generation mechanism configured to:  
3 determine whether load operations that take place during transactional  
4 execution need to be monitored;  
5 generate monitored load instructions for load operations that need to be  
6 monitored; and to  
7 generate unmonitored load instructions for load operations that do not  
8 need to be monitored.

1 16. (Original) The apparatus of claim 15, wherein the instruction  
2 generation mechanism is configured to determine whether a load operation needs  
3 to be monitored by examining a data structure associated with the load operation  
4 to determine whether the data structure is a “protected” data structure for which  
5 loads need to be monitored, or an “unprotected” data structure for which loads do  
6 not need to be monitored.

1 17. (Original) The apparatus of claim 15, wherein the instruction  
2 generation mechanism is configured to determine whether a load operation needs  
3 to be monitored by determining whether the load operation is directed to a heap,  
4 wherein loads from the heap need to be monitored and loads from outside the  
5 heap do not need to be monitored.

1           18. (Original) The apparatus of claim 15, wherein the instruction  
2     generation mechanism is configured to determine whether a load operation needs  
3     to be monitored by allowing a programmer to determine if the load operation  
4     needs to be monitored. ♡

1           19. (Original) The apparatus of claim 14, wherein the execution  
2     mechanism is configured to determine whether the load instruction is a monitored  
3     load instruction by examining an op code of the load instruction.

1           20. (Original) The apparatus of claim 14, wherein the execution  
2     mechanism is configured to determine whether the load instruction is a monitored  
3     load instruction by examining an address associated with the load instruction to  
4     determine whether the address falls within a range of addresses for which loads  
5     are monitored.

1           21. (Original) The apparatus of claim 20, wherein the execution  
2     mechanism is configured to examine the address by comparing the address with  
3     one or more boundary registers.

1           22. (Original) The apparatus of claim 20, wherein the execution  
2     mechanism is configured to examine the address by examining a Translation  
3     Lookaside Buffer (TLB) entry associated with the address.

1           23. (Original) The apparatus of claim 14, wherein if an interfering data  
2     access from another process is encountered during transactional execution of the  
3     block of instructions, the execution mechanism is configured to:  
4         discard changes made during the transactional execution; and to  
5         attempt to re-execute the block of instructions.

1           24. (Original) The apparatus of claim 14, wherein if transactional  
2 execution of the block of instructions completes without encountering an  
3 interfering data access from another process, the execution mechanism is  
4 configured to:  
5           commit changes made during the transactional execution to the  
6 architectural state of the processor; and to  
7           resume normal non-transactional execution of the program past the block  
8 of instructions.

1           25. (Original) The apparatus of claim 14, wherein an interfering data  
2 access can include:  
3           a store by another process to a cache line that has been load-marked by the  
4 process; and  
5           a load or a store by another process to a cache line that has been store-  
6 marked by the process.

1           26. (Original) The apparatus of claim 14, wherein the cache line is load-  
2 marked in level 1 (L1) cache.

1           27. (Currently amended) An computer system that selectively monitors  
2 load instructions to support transactional execution of a process, comprising:  
3           a processor;  
4           a memory;  
5           a start transactional execution mechanism within the processor configured  
6 to start a transactional execution of a block of instructions in a program, wherein  
7 | starting the transactional execution involves executing an explicit performing an  
8 instruction implemented in hardware to start the transactional execution;  
9           an execution mechanism within the processor;

10            wherein the execution mechanism is configured to support the  
11    transactional execution, and wherein changes made during the transactional  
12    execution are not committed to the architectural state of a processor until the  
13    transactional execution successfully completes;  
14            wherein upon encountering a load instruction during transactional  
15    execution, the execution mechanism is configured to,  
16                            determine whether the load instruction is a monitored load  
17                            instruction or an unmonitored load instruction by analyzing the  
18                            load instruction,  
19                            if the load instruction is a monitored load instruction, to  
20                            perform a corresponding load operation, and to load-mark a cache  
21                            line associated with the load instruction to facilitate subsequent  
22                            detection of an interfering data access to the cache line from  
23                            another process; and  
24            if the load instruction is an unmonitored load instruction, to perform the  
25    corresponding load operation without load-marking the cache line.